LIBRARY
TECH... REPORT SECTION
NAVAL ... SCHOOL
MONTEREY, CALIFORNIA 93940

NPS55Gv75101

NAVAL POSTGRADUATE SCHOOL

Monterey, California

MULTITYPE MULTIPROGRAMMING: PROBABILITY MODELS AND NUMERICAL PROCEDURES

Donald P. Gaver
and
George Humfeld

October 1975

Approved for public release; distribution unlimited.

Prepared for:

Defense Communications Agency
...ton, Virginia 22070

FEDDOCS
D 208.14/2:NPS-55Gv75101

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Isham Linder
Superintendent

Jack R. Borsting
Provost

The work reported herein was supported in part by the Defense Communications Agency, Joint Technical Support Activity, and in part by the National Science Foundation.

Reproduction of all or part of this report is authorized.

Prepared by:

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NPS55Gv70101
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE *(and Subtitle)*: Multitype Multiprogramming: Probability Models and Numerical Procedures
5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Donald P. Gaver and George Humfeld
8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: AG 476: Proposal No. P3P1823
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communication Agency, Reston, Virginia & National Science Foundation, Washington, D. C.
12. REPORT DATE: October 1975
13. NUMBER OF PAGES:
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:
15. SECURITY CLASS. *(of this report)*: Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT *(of this Report)*: Approved for public release; distribution unlimited.
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*:
18. SUPPLEMENTARY NOTES:
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*: Computer systems; Numerical methods; Queueing models
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*:

This report describes mathematical models for use in evaluating the performance of complex computer systems. Numerical methods are given to compute operating characteristics of systems when two types of jobs are present.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

MULTITYPE MULTIPROGRAMMING:

PROBABILITY MODELS AND NUMERICAL PROCEDURES

Donald P. Gaver*

George Humfeld

1. Introduction

It has long been recognized that a multiprogramming computer system may be represented by some form of cyclic queueing model, and many papers have been, and continue to be, written on this theme; see Gaver [4], Buzen [2], Baskett [1], Chandy [3]. Of considerable influence in this area was the paper by Gordon and Newell [6] which effectively demonstrated a simple structure for the joint stationary distribution of numbers of program segments (jobs) present at each of several servers: e.g. one or more CPUs and several disc drives. See also the work of Whittle [10] and of Kingman [7] in this connection. The basic assumption that has been made is that specification of a total number, J, of jobs actively being processed is given, and that a random vector $(N_{CPU}, N_1, N_2, \ldots, N_C)$ denotes the number of jobs at the CPU, and also the number of jobs at each peripheral processor. Once a job quits any processor it reports to another with a given routing probability, independently of previous history. In short, the system is modelled as a vector-state Markov process--one that is nearly explicitly solvable owing to the special structure of the equations of probability balance. Recognition of literal deficiencies in the above models have led to changed formulations. For instance, Baskett and co-workers [1] have noted that "processor sharing," i.e. a limiting form of time

*Research sponsored in part by the Joint Technical Support Activity of the Defense Communications Agency, and in part by NSF under grant number AG476 at the Naval Postgraduate School.

slicing, preserves the Markov property when jobs with several burst time distributions are simultaneously present at a server; again a convenient algebraic form of solution is available from which numerical values may be obtained to be used for assessing configurations and suggesting changes. Other studies with the goal of reducing dependence upon Markov-type assumptions (e.g. exponential burst times) have also been made, cf. Gaver and Shedler [5], and Reiser and Kobayashi [8].

The purpose of the present paper is to study systems in which different job types are present at the various servers and are processed according to a variety of scheduling disciplines, in particular in "first-come, first-served" order. Our approach is numerical, and is applied to an example which is also analyzed by means of the Gordon and Newell (GN) model. Using a plausible method of fitting the latter provides a reasonably satisfactory approximation to the "true" multitype system parameters.

2. Multi-Job-Type Markov Model Formulations and Solution Methods

In this section we present a model that explicitly recognizes the presence of jobs of different types (i.e. requiring different resources) in a multiprogramming computer system. Furthermore, the model does not specify a particular, convenient, order of service at the various processors, but rather allows that order to be specified within limits, in order to allow the effects of scheduling to be studied. For instance, one can permit jobs to be handled "first-come, first-served" at all, or a subset of processors (perhaps just at the peripherals); on other processors, processor sharing or some other form of priority may be the scheduling procedure of choice. In this paper we concentrate on "f.c.,f.s." for definiteness.

In order to carry out the above analysis it becomes necessary to (a) describe system states, (b) represent the equations of probability balance in terms of those states, (c) solve the equations of (b) in order to yield useful numbers, and (d) combine the resulting numbers into measures of system performance. In order to describe our procedure we shall deal with simple examples.

2.1 Assumptions

The system of interest consists of a single CPU and $C = 2$ peripheral processors (disc drives). The core capacity is J jobs, J considered fixed. Programs of two types inhabit the system: J_1 of type 1, and J_2 of type 2. The types exhibit different burst time and routing behavior: the rate of CPU burst completion for type i is λ_i, and of peripheral or channel completion is

μ_{ij} for type i on channel or device j. The probability that type i request peripheral processor j, abbreviated PPj upon completion of CPU burst is α_{ij}, and $\sum_j \alpha_{ij} = 1$. Note that the above structure still involves Markov-like assumptions, for burst times are exponentially distributed (memoryless) within types. But clearly a simple compilation of numbers present at CPU and peripherals does not define a Markov process unless special service rules are followed. In particular, f.c., f.s. rules do not permit such a simplification, for the exact order of the types at each server must be known. A convenient method of state description follows.

2.2 State Description: System of CPU and Two Peripherals

The system state is given by

(i) the number of jobs of each type present at each processing unit,

(ii) the identity (type) of job in process at each processor,

(iii) the order of the jobs enqueued at each processor.

A system state can be specified as a $J + 2 \times 1$ (row) vector. The first J components of the state vector are, stored in some order, J_1 "ones" and J_2 "twos" representing the job segments of type 1 and type 2. The $J + 1^{\underline{st}}$ component enumerates the jobs at peripheral processor 1, and the $J + 2^{\underline{nd}}$ component enumerates those at PP2. If the $J + 1^{\underline{st}}$ component equals N_1, and the $J + 2^{\underline{nd}}$ is N_2 then, reading from left to right, the first N_1 components of ISTATE, the state vector, are the jobs present at PP1, the second N_2 are those at PP2, and the remaining $J - N_1 - N_2$ jobs are present at the CPU. Reading from left to right in ISTATE

the jobs at any processor are listed in the reverse of their order of arrival (the line forms to the left), and hence, in the case of f.c., f.s. discipline, in the reverse of their service order at that processor. For a single processor at each point, the right-most job is the one in service, and the others are enqueued.

An example follows.

Example 1. $J_1 = 3$, $J_2 = 4$, $J = 7$.

ISTATE = {2,1,2,1,2,2,1,3,2}

2	1	2	1	2	2	1	3	2

$N_1 = 3$ in PP1, (right most in service) $N_2 = 2$ 2 in PP_2 2 in CPU N_a = number at PP1 N_b = Number at PP2

There are three jobs at PP1, a 2 is in service followed by a 1 and a 2; there are two jobs at PP2, a 2 is in service followed by a 1; there are two at the CPU, a 1 is in service followed by a 1.

Example 2. Same as above but with the following arrangement of jobs.

ISTATE = {2,1,2,1,2,2,1,0,0}.

since components $J+1$ and $J+2$ are zero, no jobs are present at PP1 or PP2, and a job of type 1 is in CPU service.

Example 3. Same as above, but

ISTATE = {2,1,2,1,2,2,1,7,0}.

Here all 7 jobs are at PP1, and the $J^{\underline{th}}$ component represents the job type in PP1 service, rather than in CPU service as before.

2.3 Sequential Numbering of States

It is convenient to re-name the states in the system by lexicographic ordering of the ISTATE vector. This is accomplished as follows for the two peripheral case; all others are treated analogously.

Example 4. $J_1 = 3$, $J_2 = 4$, $J = 7$.

The smallest (lexicographically speaking) vector representing a possible state is seen to be

$$\{1,1,1,2,2,2,2,0,0\},$$

meaning that all jobs are at the CPU, with the 2's lined up ahead of the ones. Next comes

$$\{1,1,1,2,2,2,2,0,i\}, \quad i = 1,2,\ldots,7,$$

then

$$\{1,1,1,2,2,2,2,1,i\}, \quad 0 \leq i \leq 6,$$

$$\{1,1,1,2,2,2,2,2,i\}, \quad 0 \leq i \leq 5,$$

$$\ldots$$

$$\{1,1,1,2,2,2,2,7,0\},$$

then

$$\{1,1,2,1,2,2,2,0,0\}$$

$$\ldots \qquad \ldots$$

$$\{1,1,2,1,2,2,2,7,0\}$$

$$\ldots \qquad \ldots$$

{1,1,2,2,1,2,2,0,0}

... ...

{1,1,2,2,1,2,2,7,0}

... ...

up to, finally,

{2,2,2,2,1,1,1,7,0}.

The above scheme, and its generalizations, has been programmed in FORTRAN. It can now be applied at the next stage to generate and solve the equations of probability balance.

3. Generation of Balance Equations

After the system states have been placed in lexicographic order we may speak of states i and j, where i and j are integers, and of p_i, the long-run probability that the system is in state i. The probabilities p_i, $i = 1,\ldots,I$ satisfy the system of probability balance equations

$$p_i \cdot \text{Rate of Transition from state } i = \sum_{j \neq i} p_j \cdot \text{Rate of Transition from } j \text{ to } i \qquad (3.1$$

Referring to the above as the $i^{\underline{th}}$ balance equation, where the sequence $\{i\}$ enumerates the states in the lexicographic order outlined, the balance equations are generated one at a time.

The rate of transition from state i is seen to be the sum of three numbers: the rate of transition of the appropriate job type from (a) PP1, (b) PP2, and (c) the CPU. If state i does not permit an occupant of a server, the particular rate is zero.

Example 5. ISTATE = {1,1,2,2,1,2,2,3,3}

in the same setup as before. The state number could be determined according to our rules, but whatever the number the rate of leaving the state depicted is simply $\mu_{21} + \mu_{22} + \lambda_2$.

The rate of transition to state i from j is obtained similarly.

Example 6. If, in the previous setup the $i^{\underline{th}}$ state is

ISTATE = {1,1,2,2,1,2,2,3,3,}.

then this state may be reached from the following states in one transition:

$$j_1 = \{1,1,2,2,1,2,2,3,4\}$$

or

$$j_2 = \{1,1,2,2,2,1,2,4,3\}$$

or (3.2)

$$j_3 = \{1,1,2,1,2,2,2,3,2\}$$

or

$$j_4 = \{1,2,2,1,2,2,1,2,3\}.$$

$j_1 \to i$ when a type 2 at PP2 proceeds to the CPU; the rate is μ_{22}, $j_2 \to i$ when a type 2 at PP1 goes to the CPU; the rate is μ_{21}, $j_3 \to i$ when a type 2 at the CPU goes to PP2; the rate is $\lambda_2\alpha_{22}$, and finally $j_4 \to i$ when a type 1 at the CPU goes to PP1, with rate $\lambda_1\alpha_{11}$.

4. Iterative Solution of the Balance Equations

The $i^{\underline{th}}$ balance equation, (8.1), may be expressed as follows

$$p_i = f_i(p_1, p_2, \ldots, p_I) \tag{4.1}$$

where I is the total number of states--typically very large. Also, the normalization condition prevails: $\sum_{i=1}^{I} p_i = 1$. The f_i may be derived by simply dividing the right-hand side of (3.1) by the Rate of Transition from state i, although other prescriptions are also worth considering. The above system represents a large system of linear equations with an exceptionally sparse matrix, as indicated by Example 6. Hence, an iterative solution is suggested, and the following variants of Gauss-Seidel iteration, [9] have proven to be effective.

Procedure 1

(a) Choose a set of non-negative initial gueues, $p_i^{(1)}$, for the probabilities p_i. Sometimes conveniently $p_i^{(1)} = I^{-1}$, although approximate solutions, e.g. those derived by Gordon-Newell models, may also be useful.

(b) Compute the first iteration

$$\begin{aligned}
q_1^{(2)} &= f_1(p_1^{(1)}, p_2^{(1)}, \ldots, p_I^{(1)}) \\
q_2^{(2)} &= f_2(q_1^{(2)}, p_2^{(1)}, \ldots, p_I^{(1)}) \\
&\ldots \\
q_i^{(2)} &= f_i(q_1^{(2)}, q_2^{(2)}, \ldots, q_{i-1}^{(2)}, p_i^{(1)}, \ldots, p_I^{(1)}) \\
q_I^{(2)} &= f_I(q_1^{(2)}, q_2^{(2)}, \ldots, p_I^{(1)}),
\end{aligned} \tag{4.2}$$

and then

$$p_i^{(2)} = q_i^{(2)} / \sum_{i=1}^{I} q_i^{(2)} \tag{4.3}$$

(c) In general,

$$q_i^{(n+1)} = f_i(q_1^{(n+1)}, q_2^{(n+1)}, \ldots, q_{i-1}^{(n+1)}, p_i^{(n)}, \ldots, p_I^{(n)}) \qquad (4.4)$$

and thence

$$p_i^{(n+1)} = q_i^{(n+1)} / \sum_{i=1}^{I} q_i^{(n+1)}, \qquad i = 1,2,\ldots,I \qquad (4.5)$$

provides the new, improved approximate probabilities. The above procedure is carried out until the difference $|p_i^{(n+1)} - p_i^{(n)}|$ is sufficiently small. Mathematical convergence properties of the above procedure have been substantiated in work with J. P. Lehoczky.

Procedure 2

This alternative procedure may also be used effectively, and is the one that gives rise to the numerical examples to follow.

(a) Same as above.

(b) Compute the first iteration

$$\begin{aligned}
p_2^{(2)} &= f_2(p_1^{(1)}, p_2^{(1)}, \ldots, p_I^{(1)}) \\
p_3^{(2)} &= f_3(p_1^{(1)}, p_2^{(2)}, p_3^{(1)}, \ldots, p_I^{(1)}) \\
&\ldots \\
p_i^{(2)} &= f_i(p_1^{(1)}, p_2^{(2)}, \ldots, p_{i-1}^{(2)}, p_i^{(1)}, \ldots, p_I^{(1)}) \\
&\ldots \\
p_I^{(2)} &= f_I(p_1^{(1)}, p_2^{(2)}, \ldots, p_I^{(1)}),
\end{aligned} \qquad (4.6)$$

and finally

$$p_1^{(2)} = \frac{p_1^{*(2)} + \max(1 - \sum_{i=2}^{I} p_i^{(2)}, 0)}{2} \qquad (4.7)$$

where

$$p_1^{*(2)} = f_1(p_1^{(1)}, p_2^{(2)}, \ldots, p_I^{(2)}). \qquad (4.8)$$

(c) In general,

$$p_i^{(n+1)} = f_i(p_1^{(n)}, p_2^{(n+1)}, \ldots, p_{i-1}^{(n+1)}, p_i^{(n)}, \ldots, p_I^{(n)}) \tag{4.9}$$

for $i = 2,3,\ldots,I$,

and finally

$$p_1^{(n+1)} = \frac{p_1^{*(n+1)} + \max(1 - \sum_{i=2}^{I} p_i^{(n+1)}, 0)}{2} \tag{4.10}$$

where

$$p_1^{*(n+1)} = f_1(p_1^{(n)}, p_2^{(n+1)}, \ldots, p_I^{(n+1)}). \tag{4.11}$$

5. Numerical Comparisons

Our techniques have been used to compute long-run operating characteristics for a system, these being for present purposes,

(i) the fraction of time spent idle at CPU, and at the individual PP units,

(ii) the expected number of jobs present at the CPU, and

(iii) the expected number of jobs present at the PP stage as a whole. We have been particularly interested in comparing our results with those obtained by fitting a Gordon and Newell cyclic model to a situation with multi-type jobs.

Comparison 1

Suppose it is known that

$\mu_{11} = 1.0$ $\quad \mu_{21} = \text{N.A.}$ $\quad \alpha_{11} = 1.0, \quad \alpha_{12} = 0$

$\mu_{12} = \text{N.A.}$ $\quad \mu_{22} = 1.0$ $\quad \alpha_{21} = 0.0, \quad \alpha_{22} = 1.0$

$\lambda_1 = 0.526$ $\quad \lambda_2 = 0.339$

Case 1.1 There is one type 1 job, and 1 type 2 job, so $J = 2$. If one fits a GN model by assuming

$$\rho_1 = \frac{\lambda_1 \alpha_{11}}{\mu_{11}}, \qquad \rho_2 = \frac{\lambda_2 \alpha_{22}}{\mu_{22}} \tag{5.1}$$

then the following system characteristics are predicted.

TABLE I

	Idleness Probabilities			Expected Number at Processor	
	GN	MM		GN	MM
PP1	0.60	0.80	PP1 & PP2	0.82	0.39
PP2	0.74	0.81			
CPU	0.23	0.06	CPU	1.18	1.61

Case 1.2 Here J = 3, both 2 type 1, and 1 type 2, and 1 type 1, s type 2:

TABLE 2

	Idleness Prob. GN	MM (1,2)	MM (2,1)		Expected Number GN	MM (1,2)	MM (2,1)
PP1:	0.54	0.87	0.71	PP1: & PP2:	1.09	0.43	0.50
PP2:	0.70	0.75	0.85				
CPU:	0.12	0.013	0.018	CPU:	1.91	2.57	2.50

Both the present examples and others unreported here indicate that the simple GN model, fitted as described, does not agree closely with our more complex (and realistic) model.

However, model fitting can be carried out in various ways, of which the following is a practical example when monitor data is used. Taking the job stream overall, continuity conditions lead to our equating net flow into and out of each peripheral:

$\lambda\alpha_j \cdot$ Fraction of Time CPU Busy = $\mu_j \cdot$ Fraction of Time PPj Busy.

Or

$$\rho_j = \frac{\lambda\alpha_j}{\mu_j} = \frac{\text{Fraction of Time PPj Busy}}{\text{Fraction of Time CPU Busy}} ; \qquad (5.2)$$

$j = 1,2,\ldots,c$, c being the number of peripheral processors. Now the GN theory furnishes the joint distribution of the number of jobs at, say, the peripherals (N_j = random number at PPj) in the form

$$P\{N_1=n_1, N_2=n_2, \ldots, N_c=n_c\} = K\rho_1^{n_1}, \rho_2^{n_2}, \ldots, \rho_c^{n_c} \qquad (5.3)$$

where $n_1 + n_2 + \ldots n_c \leq J$. Consequently we can calculate the idleness probabilities and expected number present at the processors explicitly in terms of the ρ_j's.

Comparison 2

Let us now suppose that observations have been made of the following multi-type multiprogramming system:

TABLE 3

PP1:	$\mu_{11} = 10$,	$\alpha_{11} = 0.75$	$\mu_{21} = 20$,	$\alpha_{21} = 0.25$
PP2:	$\mu_{12} = 20$,	$\alpha_{12} = 0.25$	$\mu_{22} = 10$,	$\alpha_{22} = 0.75$
CPU:	$\lambda_1 = 15$		$\lambda_2 = 25$	

Case 2.1 There is one type 1 job and one type 2 job (abbreviated (1,1)), and thus $J = 2$. If one observes (monitors) the system for a long time one will observe

$$\frac{\text{Fraction of time PP1 Busy}}{\text{Fraction of time CPU Busy}} = 0.81$$

and (5.4)

$$\frac{\text{Fraction of time PP2 Busy}}{\text{Fraction of time CPU Busy}} = 0.84$$

The latter figures are obtained by computing the long-run idleness probabilities of the processors by means of our iterative procedure applied to the full set of balance equations, i.e. the full-scale MM model.

Now compare the system characteristics obtained by thus solving the correct model (MM), and those obtained by applying GN, using $\rho_1 = 0.81$, and $\rho_2 = 0.84$; the figures appear in the top-most panel of Table 4. Perhaps not surprisingly, the agreement seems usefully good.

TABLE 4

	Idleness Probabilities			Expected Number	
	GN	MM		GN	MM
	J=2	(1,1)		J=2	(1,1)
PP1	.54	.52	PP1 & PP2	1.22	1.17
PP2	.53	.50			
CPU	.44	.41	CPU	0.78	0.83

	GN	MM		GN	MM	
	J=3	(1,2)	(2,1)	J=3	(1,2)	(2,1)
PP1	.45	.54	.34	PP1 & PP2 1.80	1.83	1.66
PP2	.43	.28	.57			
CPU	.32	.34	.28	CPU 1.20	1.17	1.34

	GN	MM			GN	MM	
	J=4	(1,3)	(2,2)	(3,1)	J=4	(1,3)	(2,2)	(3,1)
PP1	.39	.56	.37	.25	PP1 & PP2 2.35	2.56	2.22	2.16
PP2	.37	.16	.35	.59				
CPU	.25	.31	.22	.22	CPU 1.65	1.44	1.78	1.84

Case 2.2 Basic setup the same, but expand the jobs by one: one type 1 and two type 2 (1,2), and alternatively two type 1 and one type 2 (2,1). Compare the exact results to those obtained from GN with the above ρ_j's and $J = 3$. Thus this case, and the one to follow, exhibit the accuracy of prediction of effects of system changes (in this case number of jobs) when the system is not a GN (simple Markov) system. The results appear in the middle panel of Table 4, and again GN agreement is satisfactory, if more so for the CPU idleness probabilities than for those of the peripherals.

Case 2.3 Still the same parameter values, but additional jobs are added, to reach a total of four. Once more the GN fit seems to steer a middle course, providing about the correct impression of the effect of adding additional jobs (an approximation to adding more core storage capacity, at least in some cases).

6. Summary

We have presented models for multitype job traffic in a simple cyclic queueing model of a multiprogramming computer system (termed MM models), and have shown how such models may yield numerical results by use of a variation of the Gauss-Seidel iteration algorithm. Lastly, we have indicated the manner in which a simple Gordon and Newell Markov model may be fitted to MM data, and have shown that satisfactory predictions of some system characteristics may be obtained thereby.

REFERENCES

[1] Baskett, F., Chandy, K. M., Muntz, R. R. and Palacios-Gomez, F., "Open, closed, and mixed networks of queues with different classes of customers," J.A.C.M., Vol. 22, No. 2, April 1975.

[2] Buzen, J., "Computational algorithms for closed queueing networks with exponential servers," Com. of ACM, Vol. 16, Sept. 1973.

[3] Chandy, K. M., Herzog, U. and Woo, L., "Approximate analysis of general queueing networks," IBM Research Report, RC 4931 (#21910), July 1974.

[4] Gaver, D. P., "Probability models for multiprogramming computer systems," J. Assoc. Computing Machinery, Vol. 14, No. 3, July 1967, pp. 423-438.

[5] Gaver, D. P. and Shedler, G. S., "Approximate models for processor utilization in multiprogrammed computer systems," SIAM J. Comput., Vol. 2, No. 3, Sept. 1973.

[6] Gordon, W. and Newell, G. F., "Closed queueing systems with exponential servers," Opns. Res., Vol. 15, 1967.

[7] Kingman, J. F. C., "Markov population processes," J. of Appl. Prob., Vol. 6, No. 1, April 1969.

[8] Reiser, M. and Kobayashi, H., "Accuracy of the diffusion approximation for some queueing systems," IBM J. of Res. and Dev., Vol. 18, 1974.

[9] Varga, R. S., Matrix Iterative Analysis, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1972.

[10] Whittle, P., "Nonlinear migration processes," Proc. 36th Session of the International Statistical Inst., 1967.

INITIAL DISTRIBUTION LIST

	Copies
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
Dean of Research Code 023 Naval Postgraduate School Monterey, California 93940	1
Library (Code 0212) Naval Postgraduate School Monterey, California 93940	2
Library (Code 55) Naval Postgraduate School Monterey, California 93940	2
C. Giammo Defense Communication Agency JTSA 11440 Isaac Newton Square Reston, Virginia 22090	5
B. Wallach Defense Communication Agency JTSA 11440 Isaac Newton Square Reston, Virginia 22090	2
P. Kiviat FEDSIM Washington, D. C. 20330	4
M. Spiegel FEDSIM Washington, D. C. 20330	1
L. Kleinrock Computer Science Department UCLA Los Angeles, California 90024	1
R. Muntz Computer Science Department UCLA Los Angeles, California 90024	1

E. Gelenbe 1
IRIA (Laboria)
Domaine de Voluceau - Rocquecourt
B.P. 5 - 78150 Le Chesnay
France

G. Shedler 1
IBM Research
San Jose, California 95100

W. Tuel 1
IBM Research
San Jose, California 95100

K. Marshall 1
P. Milch 1
M. Thomas 1
R. Butterworth 1
P. A. W. Lewis 1
D. Gaver 15
Code 55
Naval Postgraduate School
Monterey, California 93940

G. F. Newell 1
Transportation-Civil Engineering
University of California
Berkeley, California 94720

U170554

DUDLEY KNOX LIBRARY - RESEARCH REPORTS
5 6853 01071142 7